FlowQoS: QoS for the Rest of Us

M. Said Seddiki[*†‡], Muhammad Shahbaz[*], Sean Donovan[*], Sarthak Grover[*],
Miseon Park[*], Nick Feamster[*], Ye-Qiong Song[†]
[*] Georgia Tech, [†] Loria, Lorraine University, [‡] Sup'Com, University of Carthage

Abstract

We describe the architecture of FlowQoS, a system that makes it easier for users in home broadband access networks to configure quality of service based on applications and devices, as opposed to obscure, low-level parameters. The central tenet of FlowQoS's design is control logic that performs application identification and uses flow-table rules to forward traffic through the appropriate rate shapers on a home router. The architecture has two components: a flow classifier, which maps application traffic to the appropriate parts of flow space; and an SDN-based rate shaper, which shapes application traffic by forwarding it through the appropriate shaped virtual links in the home gateway. This paper describes the high-level architecture of FlowQoS, as well as our current implementation.

Categories and Subject Descriptors:
C.2.3 [Computer-Communication Networks] *Network Operations*: Network Management

Keywords: Software Defined Networking (SDN); Home Networks; Bandwidth management; Quality of Service (QoS)

1 Introduction

Managing QoS in home networks is challenging. Home users expect good performance for data, voice, and video, all with reasonable Quality of Service (QoS). Guaranteeing good QoS for these applications involves configuring priorities and facilitating sophisticated per-flow, application-based QoS to prevent one Internet application from degrading overall performance when it competes for bandwidth with other applications. ISPs may have trouble satisfying these requirements because doing so requires deploying specialized equipment in home networks. On the other hand, the home user has a limited knowledge on how to manage QoS for multiple traffic flows.

Several mechanisms to provide QoS in home networks have been proposed and implemented [2, 3, 4, 6, 8], but these mechanisms have not yet been deployed in broadband access networks. Current home routers generally have limited computational resources, so performing application classification may be prohibitive. Users may also have difficulty configuring QoS functions that are complicated and obtuse, rather than based on specific applications or devices. One possible solution to address these issues is to delegate QoS functions to separate control logic that allows a user to specify QoS policies at a higher level of abstraction. Software Defined Networking (SDN) [5] can facilitate such a redesign, by separating the network control plane from the forwarding plane. A control application could run directly on the router itself as a separate program (in the case where the router is powerful enough to run it), or on a separate device, either inside the home or from a remote location.

In this paper, we present FlowQoS, a system that performs per-flow, application-based QoS by delegating application identification and QoS configuration to an SDN controller. In FlowQoS, the user of the broadband access network simply specifies the high-level applications that should have higher priority (*e.g.*, adaptive video streaming, VoIP), and the FlowQoS controller performs the appropriate application identification and QoS configuration for both upstream and downstream traffic to implement the user's preferences. For each flow, FlowQoS performs on-the-fly application identification. It also installs rules in the data plane that forward individual flows according to user-specified priorities for those applications. Our system creates links in a virtual topology in the home router, configures each of these links with a user-specified rate, and assigns flows to these links to provide rate shaping per application.

2 FlowQoS Architecture

Figure 1 shows the high-level architecture of FlowQoS. Users specify the maximum allowed bandwidth for specific high-level applications in the home network using a web configuration tool. This tool then creates the configuration for the rate shaper, which has two main components described below: the flow classifier and the SDN-based rate controller.

Flow classifier: This component maintains a lookup table where the key is a flow tuple consisting of the source IP address, destination IP address, protocol, source port, and destination port. This component uses two modules to perform traffic classification. The first classifier performs early application identification of HTTP and HTTPS traffic, and the second classifier performs application identification for other flows. The application traffic on ports 80 and 443 (*i.e.*, HTTP and HTTPS, respectively) are handled by FlowQoS's DNS-based classifier. It performs more fine-grained classification; many classifiers would otherwise classify many types of application traffic on these ports simply as "web traffic". This module maintains a table that it builds using the DNS responses that the switch forwards. The table includes the A or CNAME record, the corresponding IP address (in the case of an A record response), and the time-to-live for the record. To classify flows based on this information, the classifier checks the A or CNAME record against a list of regular expressions, each of which corresponds to an application type. Because the sender initiates a DNS request before the corresponding TCP

HotSDN'14, August 22, 2014, Chicago, IL, USA.
ACM 978-1-4503-2989-7/14/08.
http://dx.doi.org/10.1145/2620728.2620766

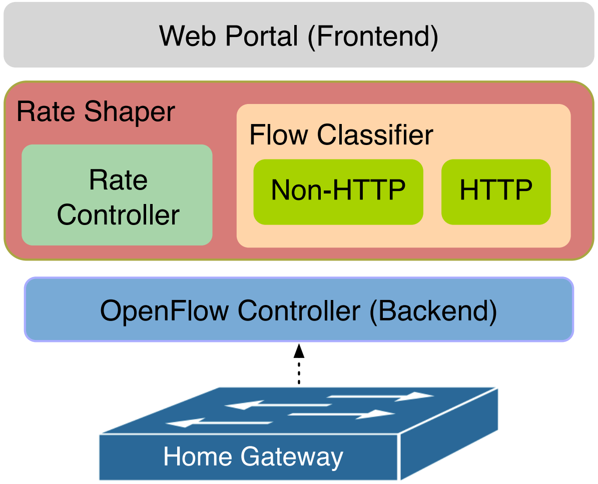

Figure 1: *FlowQoS architecture*

connection, the classifier can associate a flow with an application before the sender even sends the first packet of the flow.

The second module uses a modified `libprotoident` library [1] to perform application layer protocol identification for flows. To classify traffic it relies on the flow tuple, as well as the first four bytes sent, first four bytes received, first payload size sent, and first payload size received. The requirement for parsing these additional fields might require the controller to see additional few packets before processing the flow, but because QoS enforcement is lazy (*i.e.*, before a flow is classified, it is forwarded on the default queue), the requirement to see additional packets is not prohibitive.

Rate controller: Based on the results of classification, the controller installs rules in the switch for that flow. These rules forward the flow's packets through the appropriate path (and, hence, queue) in the switch. Once the association between flows and applications is recognized, FlowQoS's SDN-based rate controller assigns each flow to the appropriate rate. In existing systems, assigning a priority to each traffic flow according to user configuration is complicated by the limitations of today's home routers: Existing home routers do not support per-flow rate control; existing mechanisms such as `tc` still require configuring virtual interfaces or tagging via `iptables`. Even the current Open vSwitch implementation for OpenWrt does not yet support the parts of the OpenFlow 1.3 specification that provide for per-flow QoS.

To overcome these limitations, FlowQoS enables per-flow QoS by instantiating a two-switch virtual topology on the home router, as shown in Figure 2. Each virtual link between the two switches corresponds to a different application group (*e.g.*, video, web, gaming). To implement rate limiting, each link has a traffic shaper (implemented with Linux's `tc` utility) that corresponds to the user-specified rate. To perform rate limiting once the classifier has identified the application type for a flow, the switch refers to its existing rules to determine which inter-switch connection corresponds to that traffic class. Only flows that need to be rate-limited will be categorized in a different manner.

When a new flow arrives at the switch, it is redirected to the appropriate flow classifier. Based on the results from classification, the controller installs OpenFlow rules into the Open vSwitch components such that the new flow is forwarded on the virtual links with the appropriate shaping parameters. Configuring `tc` on virtual inter-switch links is conceptually similar to what OF-config [7] might enable if Open vSwitch supported the flow-based QoS functions outlined in OpenFlow 1.3. Our dual-switch topology is a workaround for the limitations of the Open vSwitch implementation.

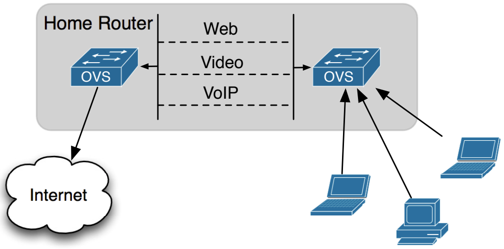

Figure 2: *The virtual switch topology that performs traffic shaping inside the home router.*

3 Implementation

We have implemented a prototype of FlowQoS on OpenWrt. We integrated Open vSwitch with OpenWrt to enable the control of an OpenWrt switch using OpenFlow. We used a Raspberry Pi [10] for the controller hardware. We implemented the control application on top of POX [9], a popular open-source OpenFlow controller. Preliminary results show that FlowQoS improves the performance of both adaptive video streaming and VoIP in the face of competing traffic. We are currently extending the system to support additional features and applications.

Acknowledgments

This research was supported by NSF awards CNS-1059350 and CNS-1261357, Fulbright Foreign Student Fellowship, and a Google Focused Research Award.

References

[1] S. Alcock and R. Nelson. Libprotoident: Traffic classification using lightweight packet inspection (technical report). University of Waikato, 2012.

[2] C. Aurrecoechea, A. T. Campbell, and L. Hauw. A survey of qos architectures. *Multimedia systems*, 6(3):138–151, 1998.

[3] J. Gozdecki, A. Jajszczyk, and R. Stankiewicz. Quality of service terminology in ip networks. *Communications Magazine, IEEE*, 41(3):153–159, 2003.

[4] D. McDysan. *QoS and traffic management in IP and ATM networks*. McGraw-Hill, Inc., 1999.

[5] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. Openflow: enabling innovation in campus networks. volume 38, pages 69–74. ACM, 2008.

[6] Meraki Traffic Shaper. `http://goo.gl/IL24ek`.

[7] OF-CONFIG 1.2: OpenFlow Management and Configuration Protocol. `http://goo.gl/1JnFWN`, 2014.

[8] PacketShaper. `http://www.bluecoat.com/products/packetshaper`.

[9] POX. `http://www.noxrepo.org/pox/about-pox/`.

[10] E. Upton and G. Halfacree. *Raspberry Pi User Guide*. John Wiley & Sons, 2012.